Unterspezifikation in der Computationellen Semantik
Hausaufgabe 3

Laura Kallmeyer

WS 2011/2012, Heinrich-Heine-Universität Düsseldorf

Aufgabe 1 (Abgabe: 31.10.2011)

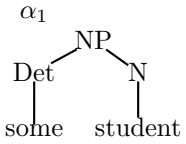
Betrachten Sie folgendes Beispiel:

(1) some student likes every course.

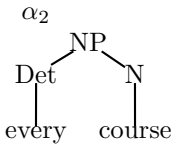
Zeigen Sie, wie sich die Semantik für die beiden Skopuslesarten unter Verwendung von Quantifying In *berechnen lässt. Geben Sie insbesondere die verschiedenen (Teil-)bäume an, die sich in der Syntax ergeben und zeigen Sie jeweils, wie sich in Abhängigkeit von der durchgeführten syntaktischen Operation deren Semantik berechnet.*

Lösung:

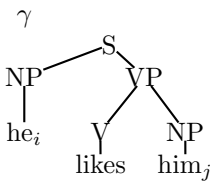
1. Zunächst werden die Bäume für *some student*, *every course* und he_i *likes* him_j bottom-up aufgebaut. Die Pronomen tragen jeweils ihren Index (Typ e) zur Semantik bei.



Semantik des NP-Baums:
$[[\alpha_1]]$
$= \lambda P.\lambda Q.some'(\lambda x.P(x), \lambda x.Q(x))(student')$
$= \lambda Q.some'(\lambda x.student'(x), \lambda x.Q(x))$



Semantik des NP-Baums:
$[[\alpha_2]]$
$= \lambda P.\lambda Q.every'(\lambda x.P(x), \lambda x.Q(x))(course')$
$= \lambda Q.every'(\lambda x.course'(x), \lambda x.Q(x))$

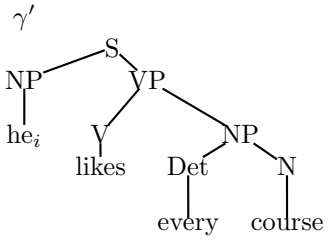


Semantik des S-Baums:
$[[\gamma]]$
$= like'(i, j)$

2. Erste Lesart *some* > *every*:

   Zunächst wird α_2 mit γ durch Quantifying-In bzgl. j verknüpft, anschließend α_1 mit dem Resultat aus der ersten Anwendung von Quantifying In.

   $Syn_{Qu-in.j}(\alpha_2, \gamma)$: Ersetzen des Pronomens mit Index j durch α_2. Ergebnis:

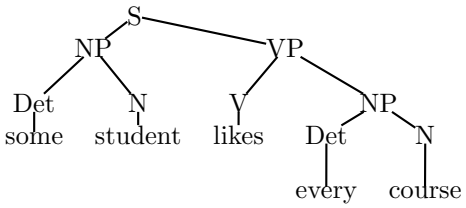


   Entsprechende semantische Operation:

$[[\gamma']]$
$= [[Syn_{Qu-in.j}(\alpha_2, \gamma)]]$
$= [[\alpha_2]](\lambda y.[[\gamma]]^{y/j})$
$= \lambda Q.every'(\lambda x.course'(x), \lambda x.Q(x))(\lambda y.like'(i, y))$
$= every'(\lambda x.course'(x), \lambda x.(\lambda y.like'(i, y))(x))$
$= every'(\lambda x.course'(x), \lambda x.like'(i, x))$

$Syn_{Qu-in.i}(\alpha_1, \gamma')$: Ersetzen des Pronomens mit Index i durch α_1. Ergebnis:

γ''



$[[\gamma'']]$
$= [[Syn_{Qu-in.i}(\alpha_1, \gamma')]]$
$= [[\alpha_1]](\lambda y.[[\gamma']]^{y/i})$
$= \lambda Q.some'(\lambda x.student'(x), \lambda x.Q(x))(\lambda y.every'(\lambda z.course'(z), \lambda z.like'(y, z)))$
$= some'(\lambda x.student'(x), \lambda x.(\lambda y.every'(\lambda z.course'(z), \lambda z.like'(y, z)))(x))$
$= some'(\lambda x.student'(x), \lambda x.every'(\lambda z.course'(z), \lambda z.like'(x, z)))$

3. Zweite Lesart *every* > *some*:

   Hier wird zunächst Quantifying-In auf α_1 und γ angewendet (mit Index i), und dann auf α_2 und das bisherige Ergebnis (mit Index j). In der Syntax ergibt sich wieder der Baum γ'', aber mit einer anderen Semantik:

   $[[\gamma'']]$
   $= [[Syn_{Qu-in.j}(\alpha_2, Syn_{Qu-in.i}(\alpha_1, \gamma))]]$

   $[[Syn_{Qu-in.i}(\alpha_1, \gamma)]]$
   $= [[\alpha_1]](\lambda y.[[\gamma]]^{y/i})$
   $= \lambda Q.some'(\lambda x.student'(x), \lambda x.Q(x))(\lambda y.like'(y, j))$
   $= some'(\lambda x.student'(x), \lambda x.like'(x, j))$

   $[[Syn_{Qu-in.j}(\alpha_2, Syn_{Qu-in.i}(\alpha_1, \gamma))]]$
   $= \lambda Q.every'(\lambda z.course'(z), \lambda z.Q(z))(\lambda y.some'(\lambda x.student'(x), \lambda x.like'(x, y)))$
   $= every'(\lambda z.course'(z), \lambda z.some'(\lambda x.student'(x), \lambda x.like'(x, z)))$